Moving Target Defense Using Live Migration of Docker Containers

by

Bhakti Bohara

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2017 by the
Graduate Supervisory Committee:

Dijiang Huang, Chair
Adam Doupe
Ziming Zhao

ARIZONA STATE UNIVERSITY

August 2017

ABSTRACT

Today the information technology systems have addresses, software stacks and other configuration remaining unchanged for a long period of time. This paves way for malicious attacks in the system from unknown vulnerabilities. The attacker can take advantage of this situation and plan their attacks with sufficient time. To protect our system from this threat, Moving Target Defense is required where the attack surface is dynamically changed, making it difficult to strike.

In this thesis, I incorporate live migration of Docker container using CRIU (checkpoint restore) for moving target defense. There are 460K Dockerized applications, a 3100% growth over 2 years[1]. Over 4 billion containers have been pulled so far from Docker hub. Docker is supported by a large and fast growing community of contributors and users. As an example, there are 125K Docker Meetup members worldwide. As we see industry adapting to Docker rapidly, a moving target defense solution involving containers is beneficial for being robust and fast. A proof of concept implementation is included for studying performance attributes of Docker migration.

The detection of attack is using a scenario involving definitions of normal events on servers. By defining system activities, and extracting syslog in centralized server, attack can be detected via extracting abnormal activates and this detection can be a trigger for the Docker migration.

DEDICATION

This work is dedicated to Ramesh Bohara and Pushpa Bohara (my parents) for teaching me not to give up. I also dedicate my work to my elder sister Prerana Bohara, who has been a strength and an inspiration. Last but not the least, to Shruti Bohara and Varun Bohara, who inspire me to be an ideal elder sister every day.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Dr Dijiang Huang for giving me the opportunity to enter the world of research. He has been a positive influence for me to push my boundaries and never settle.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr Adam Doupe and Dr Ziming Zhao for their insightful comments and encouragement.

I must express my immense gratitude to members of SNAC Lab for guiding me whenever at each step of thesis. I could not have completed it without their support.

Most importantly, none of this could have happened without my family. My parents, my sisters and brother. Every time I was ready to quit, you did not let me and I am forever grateful. This work is testament to your unconditional love and encouragement.

Getting through my thesis required more than academic support, and I have my friends at ASU, who are my family here in the USA for listening to me and tolerating me over past two years.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Live migration has been a topic of interest for mostly reasons like load balancing, maintenance of systems. It has been also introduced for security in moving target defense scene in dynamic network category. As the industry is rapidly adapting to Docker for applications, in this thesis I introduced Docker migration as a part of moving target defense. Docker containers have different file structure than virtual machines. They are much lighter and hence will provide a lesser downtime compared to virtual machine migration.

I will also discuss performance metrics for every step in the migration of the containers.

The detection of attack in done by event tracing. I am defining system information in audit policy which will identify which traffic is abnormal vs normal. The logs are collected at a centralized server using rsyslog configuration. Once combined, all the abnormal behavior gives a time line of events that are threatful and contribute towards much higher impactful attack.

CHAPTER 2

BACKGROUND

Moving Target Defense

Today information technology systems have addresses, software stacks and other configurations remaining unchanged for a long period of time. This paves way for malicious attacks in the system from unknown vulnerabilities. The attacker can take advantage of this situation and plan their attacks with sufficient time. To protect our systems from this threat, Moving Target Defense project is introduced where attack surface is dynamically changed making it difficult to be strike.

US Department of Homeland Security defines Moving Target Defense as "Moving Target Defense (MTD) is the concept of controlling change across multiple system dimensions in order to increase uncertainty and apparent complexity for attackers, reduce their window of opportunity and increase the costs of their probing and attack efforts. MTD assumes that perfect security is unattainable" [2]

The techniques of moving target defense depends on which static configurations like addresses, names, software stacks, networks are changed over time. In this section, I will be discussing the majority of them.

The Moving Target Defense techniques can be broadly classified into the types as shown in the Table 1

Table 1 : Techniques in Moving Target Defense[3]

Type	Description
Dynamic Runtime Environment	Techniques that change the environment presented to an application by the operating system (OS) during execution dynamically.
Dynamic Software	Techniques that change application's code dynamically. The change can include modifying the program instructions, their order, their grouping, and their format.
Dynamic Data	Techniques that change the format, syntax, encoding, or representation of application data dynamically
Dynamic Platforms	Techniques that change platform properties (e.g., central processing unit (CPU), OS) dynamically. This can include the OS version, CPU architecture, OS instance, platform data format, etc.
Dynamic Networks	Techniques that change network properties including protocols or addresses dynamically

Docker

Docker is an extension of Linux Containers (LXC). It is a unique kind of lightweight, application-centric virtualization that drastically reduces overhead and makes it easier to deploy software on servers. Docker allows a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. It eliminates the problem of "works on my system".

In order to understand difference between Virtual Machines and Docker we need to know that virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot taking upto 2-3 minutes

depending on their configuration. Containers are an abstraction at the application layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.On same hardware investment , we can have more containers than VMs .

On the security front, Docker depends on the security of the host machine. If there are inherent vulnerabilities in the host machine, then docker can get vulnerable to attacks.

Checkpoint Restore in User Space

Check point restore in user space is an open source project created by Pavel Emelyanov [17]. It was created with the goal of live migration of containers. It implements the checkpoint/restore functionality in the user space. This helped me to build the proof of concept of performing live migration for Moving Target Defense, It freezes an running application as a collection of files on disk in form of images. These images are used to restore the application from checkpoint state on any host that has similar configuration of docker containers.

Related Work

The following literature review shows the recent advances in the field of Moving Target Defense.

Carvalho, Marco, and Richard Ford. “Moving-target defenses for computer networks.” (IEEE Security & Privacy 2 (2014): 73-76) [4] showed brief overview of Moving Target

Defense and its potential application to computer network security. In traditional system, due to static network architecture, attackers have infinite amount of time to learn about the infrastructure and its potential vulnerabilities to achieve their goals. This gives attackers an unfair advantage compared to defenders. In such cases defense is mainly reactive and no proactive. This is state of the art in network defense. Learning from attacks and updating systems can increase the cost of infrastructure and still be vulnerable. New techniques are explored to mitigate the effort and cost asymmetry between attackers and defenders. Moving target defense does not focus on shielding but on manipulation and control of targets.

MTD could choose various ways of defense. One of them is to periodically change part of OS to change a critical service It could restructure a software defined network's topology by changing address space to defend against brute force. MTD is proactive and works without feedback. It can adapt at a slower pace than reactive defense as it increases network complexity. Reactive MTD can make changes in system when they receive trigger from a security sensor. Mostly MTD system use adaptive response which takes sensors triggers and events to change their behavior and hence moving between proactive states. One of the most citied example of MTD based on use of ASLR, that randomly arranges address space to make it difficult to exploit.

Using MTD resilience systems can be built that are able to detect and recover from security events and develop ways to evade these attacks. Moving target defense can increase system complexity from a defensive angle. Enabling a system to move and adapt for defense that makes it difficult to track and maintain. Intelligent attackers can also adapt and completely

change their operating procedures to attack the defense model. Attacker- defender co-evolution is thus an important challenge.

Evans, David, Anh Nguyen-Tuong, and John Knight. “Effectiveness of moving target defenses.” Moving Target Defense. Springer New York, 2011. 29-48 [5] described in their paper the diversification of the semantics of the lower level program. Though the advantage is that it can be done automatically independent of behavioral specification, the limitation is that these diversification is that they can change behavior only for exploits of undefined semantics. Therefore the system can be still affected by application level attacks and is immune from the code injection and memory corruption attacks. The other type of diversity alters at application level. This highly depends on the clear understanding of the application’s behavior and ensure that defense from attack is possible without compromising the functionality. This leads to high manual load to produce variants. The paper describes three common types of automatic diversity techniques to disrupt exploits. Address space randomization to randomize the location of objects in memory so that attacks that exploit the information of the address fail. Instruction set randomization is technique for thwarting code injection attacks by obscuring instruction set of target. Attacks will not be possible if a code is being injected into target application without knowing the target’s instruction set. Data randomization is altering how data is stored in memory and is a type of low level diversification.

The model consists of two players, attacker and defender. Defender has to provide service with high reliability and performance whereas attacker has to exploit the server. Assumption is that service has at least one vulnerability which the attacker knows. They consider time te between starting to launch the exploit and system compromise.

Considering that instead of the server, the defender generates the key k. If attacker determines the key, then it can construct an exploit ak to achieve desired compromise. The defense succeeds when all attacker's exploit is in target class of attack input and attackers has not learned enough information about key to exploit ak. The attacker probes the server to get information about the key. This increases over time and the attacker learns more about the target service. At some point the attacker is able to break in the system. If dynamic diversity is applied, then service now has new key for every period and hence the attack is disrupted. The goal is to understand what type of diversity can be disrupted with such strategy.

Effectiveness of the moving target defense depends on the attacker's capabilities, resources and strategy. One of them is circumvention attacks, where if the attacker finds any exploit that does not depend on the properties that could have been altered by diversification. Return to libc attack, incomplete randomization are common examples. In Deputy attacks, confused deputy attack attacker finds a benign program in a malicious way. Example of this attack is partial overwrite attack that modifies the least significant byte of an address to the program's control is transferred to a targeted function. Dynamic diversity does not provide any advantage for deputy attacks, as the attacker is exploiting actual transformation. Brute force attacks in attempting all the randomization keys until an exploit is found. In Entropy reduction attacked like a NOP sled, which is used in buffer overflow are used to overcome uncertainty in the program. Dynamic diversity has modest benefit against a brute force or entropy reduction attack, as the maximum impact on the attacker is changing the random search with replacement. A javascript head spraying attack is where an attacker uses Javascript code executed by browser to allocate large number of objects

in a heap each of which includes NOP sled in order to increase the likelihood that a jump to a randomized address will reach one of the copies in memory of the exploit code.

Probing attack attempts overcomes diversity defense by using probe packets. There are designed only for obtaining information about the target, rather than produce malicious behavior. Buffer overflow attacks expose randomized addresses in memory and takes advantage of the strncpy library functions. Dynamic diversity could be useful in case of probe attacks if the time between the probe and exploit is long. Frequent re-randomization can be expensive but can be done in some scenarios. Incremental attacks are form of probing attacks where more than one successful probe is required to obtain sufficient information to construct the exploit. Dynamic diversity is most promising against incremental attacks since attacker prepares itself with information before randomization.

To improve effectiveness of diversity defenses, two approaches can be used, the first is composition that will increase the value if re-randomisation and the second is for attacker to simultaneously compromise multiple variants.

R. Zhuang, S. Zhang, A. Bardas, S. A. DeLoach, X. Ou and A. Singhal, "Investigating the application of moving target defenses to network security"[6] show Logical Mission Model to capture abstract view of the physical network. The driver is the Adaption Engine that orders random adaptions to the network at random intervals. These adaptions are implemented by Configuration Manager which controls of configuration of Physical network. In addition we have Analysis engine that takes real time events from the physical network, current configuration from the Configuration Manager. Logical security model captures network state along with security state. Goal is to capture system's security goals while capturing vulnerability using a novel method called Conservative Attack Graph.

Resource mapping system is implemented as a system communication enforcement component. It interacts with configuration manager and pushes the up to date location information of the various resources to corresponding RMS. Assumption is that each mission critical role is executed on a unique virtual machine. Limitation is when the attackers compromises a critical role or a VM. In this case roles (network services) with which compromised role initiates communications can be easily located and attacked. Attacker must follow exact communication pattern defined by the Logical Mission Model. Adaptive Engine objective is to produce effective configuration that are significantly different in costs of adaption or maximizing the entropy. They should be functionally correct and consistent. Using intelligent adaptions with randomization allows MTD to effectively mitigate unpredicted attacks as well as mask the actions of the control systems. Since the Engine is the main decision making component of MTD , It should be able to control IP addresses, resources , firewalls between services.

Analysis engine proposes a conservative attack graph that reduces size of the state model. Topology of the graph is partially derived from logical mission model. In normal operations users login through authorizer node and interact with planner node. CAG captures these logical path and can be now viewed as a state transition system describing activities involved to move from one state to the next.

The paper quantifies impact of MTD systems on computer networks. End goal is to develop objective models to predict the effectiveness. Cost of the MTD system, the detectors, overhead of adapting the system and real time data collection is a major issue. Design of the RMS system is also an major issue where it has made several assumptions. Future work is to build the system using open stack and work on the major issues as described before.

Jia, Quan, Kun Sun, and Angelos Stavrou. “Motag: Moving target defense against internet denial of service attacks.” [7] Computer Communications and Networks

This paper attempts to solve the problem of distributed denial of service attack by proposing MOTAG, Moving Target Defense against Internet Denial of Service Attacks. It offers resilience for authorized and authenticated clients of security sensitive services such as online banking. It employs a layer of secret moving proxies as a medium for all communications between client and servers. The network level filters only allow traffic to valid proxy intermediate nodes to reach the server.

Proxy nodes in MOTAG have two characteristic, one that they are secret, their IP address are not disclosed to public. Second they are moving, if an active proxy is attacked it is replaced by another node at different location. These help in mitigating brute force attacks and allow us to discover malicious insiders and isolate them. This is done by shuffling client assignment to new proxy nodes. The proposed algorithm accurately estimates number of insiders, and adjust the assignment accordingly.

The solution does not rely on global adoption of Internet routes nor depends on resource abundant overlay network and hence avoiding bandwidth attacks and fault tolerance. This is taken care by the secret proxies and thereby reducing the costs of deployment while offering DDOS protection.

The proposed system only target security sensitive online services against network flooding attacks. General purpose web services are not considered here. Assumption is to have a large pool of backup proxies that attacker is not capable of attacking together at once. High bandwidth attackers are assumed that are capable of simultaneously overloading multiple

machines. If attacker is aware of MOTAG mechanism then they can flood the authentication channel hence blocking legitimate clients.

MOTAG uses DNS registration for associating application domain name with IP address. Application servers maintain a dedicated interface with proxies to signal in case of attack. For communication capability token is used and for a session, proxy should receive same capability token from both parties in communication. Proxy and application communicate via a light weighted authenticator that can be dynamically altered.

If one proxy node is attacked, it will be shut down and a new proxy node will be activated as a replacement. This is fast light weighted operation are proxies are simple traffic indirection logic and stores no client state. All clients are reassigned. Session information are stored on application servers. Client reallocation is done via client to proxy shuffling. Authentication server assures accessibility to the moving target defense. Every client has to pass the authentication before being assigned to a proxy. As authentication server is publicly addressed it can be target of distributed flood attacks. Proof of work approach are suitable for protecting client authentication in packets that are infrequently sent to server.

Proxies are divided into shuffling proxy and serving proxies. When attack shuffling proxies will be replaced and associated clients are flushed and reassigned. Algorithm used is greedy shuffling algorithms that is recursively called to assign clients to proxies. In security analysis it has been proved that the designed approach MOTAG is invulnerable to scanning attacks (brute force). Malicious insider can pose a serious threat and MOTAG can have these insiders quarantined in few rounds. It also provides resistance to compromised proxies using light weight authenticators to verify proxy to server traffic. The approach produces two aspect of overheads, i.e. proxy based communication indirection and client

to proxy shuffling. Nevertheless the framework can protect DDOS attacks by using hidden proxies.

Han, Yujuan, Wenlian Lu, and Shouhuai Xu. “Characterizing the power of moving target defense via cyber epidemic dynamics.”[8]

The paper studies how to characterize the power of moving target defense in cyber epidemic dynamics. The paper defines two novel approach that are applicable when using MTD to achieve certain defense goal. The first one is to obtain maximum portion of time the system can afford to stay in an insecure configuration, without considering the cost of deploying the MTD. The second is minimum cost when system can stay in insecure configuration for a predetermined time. Cyber epidemic dynamic models use graph theoretic abstraction to represent attack-defense structure and use parameters to represent attack and defense capabilities.

Network based MTD technique: The basic idea is to dynamically regulate which network address space can access the services of other address space. By randomizing IP address we can defend machine against direct attacks to target computers.

Hosts based MTD technique: Instruction level randomization: This aims to randomize instruction of each process so that attacker cannot inject malicious code. Code level randomization: It offers defense against code reuse attacks by substituting instructions, inserting NOPs and reallocating registers. ASLR defends against code injection attacks by randomization of memory layout of a program. Application level randomization: N version programming that protects dynamically using different implementation of same function. Cryptography can avoid this single point of failure because the key has been split in t pieces

that will not cause exposure of keys. Proactive cryptography can render compromised pieces of key as useless even if reconstructed.

Instrument based MTD technique: Here honeypot like technique can be used to capture new attacks. Defender can dynamically change the IP address monitored by honeypots.

There exists as an attacker, victim relation that can be depicted by a graph called attack-defense structures. MTD provides proactive defense. The approach in the paper offers algorithms for deploying MTD where time that the system can afford to stay in undesired configuration can be maximized and the cost of launching MTD can be minimized. The study assumes that the attacker defense structure with parameters. It gives no information how these parameters are obtained. The paper also does not allow attacker to choose when to use specified configuration as in the real world it is important to give attacker to give choice of configuration.

Hong, Jin B., and Dong Seong Kim. "Scalable security models for assessing effectiveness of moving target defenses." [9]

In this paper the authors incorporate MTD technique in security modeling using Hierarchical Attack Representation Models (HARMs). They classify the moving target defense techniques into three major categories, shuffle (rearrange system setting at various levels), diversity (provide equivalent functions but different implementation) and redundancy (Multiple replica of services or nodes to make multiples of same data). Their idea is to use attack representation model (ARM) for assessing effectiveness of MTD techniques. Number of path increases exponentially in ARMs as the number of nodes grow. To address this problem a hierarchical attack representational model is used for security

analysis. It is more scalable than attack graphs and can adopt any ARMs in its layers. It is adaptable to single layer changes in networked systems.

The contributions of the paper are to "Incorporating and analyzing the effectiveness of the MTD techniques using the HARM, Applying the MTD techniques to secure attack paths, using the Ims to deploy the MTD techniques in scalable and effective manners.". The paper also discusses the modelling methods for incorporating the MTD techniques in HARM.

Assumption: Even a fixed system can have changes in ARM by various reasons such as changes in topology system updates and patching of vulnerabilities. Attacker exploits vulnerabilities of OS. Components can be changed frequently. The authors build the system on these constraints using virtual machines.

Security analysis of MTD techniques is mostly through deployment of the methods to weak points and capturing possible changes to the network. To incorporate shuffling method, a live virtual machine migration is performed. It is proved in results that it take O(N) time complexity to update changes using HARM. To incorporate diversity, for a VM they apply diversity on OS layer and have an attack tree on lower layer .Diversity changes attack surface by forcing the attacker to use different exploits.

The authors validate the models of deploying MTD using a real system, considering vulnerable vulnerabilities and then evaluation of MTD techniques with various security metrics. Lastly they also provide optimization of important measures with respect to MTD techniques.

Debroy, Saptarshi, et al. "Frequency-Minimal Moving Target Defense using Software-Defined Networking."[10]

The paper proposes a novel approach “frequency minimization and consequent location selection of target movement across heterogeneous virtual machines based on attack probability, which in turn minimizes cloud management overheads.”[10] It tends to solve the MTD protection design issues with SDN enabled platform. The proposed system allows dual mode operation, proactive migration of target application and reactive migration in loss of availability (LOA) attack. This is achieved by frequency minimization and location selection of target movement across heterogeneous virtual machines.

To implement the approach, the authors compare attack probability and migration interval selection for different cost budget. If the attacker has high budget for attack, the migration will be more frequent and probability will be low. To counter the LOA attacks the paper computes a candidate depending upon storage capacity, network bandwidth and attack history. Once the target is migrated to new VM all the users are redirected to chosen destination VM using Openflow.

The scheme is evaluated over GENI infrastructure. The feature JUST IN TIME news feed application provides unique target usecase with cloud service. The strategy is compared to static schemes to prove increase in optimism. System model consists of Openflow controller and VMs. Optimal migration frequency will be such that its is not too infrequent to make system vulnerable to cyber-attacks and under this predicate optimization problem is mathematically formulated. The paper successfully proves performance benefit migration process and performs tradeoff between cost of migration and benefits of protection. They however are not able to provide a plan for minimization of cloud cost of MTD based solution.

Vadlamudi, Satya Gautam, et al. “Moving Target Defense for Web Applications using Bayesian Stackelberg Games.” arXiv preprint arXiv:1602.07024 (2016). [11]

This paper presents a way to find effective switching strategies by modeling system using Bayesian Stackelberg games. The administrator is the leader and the hackers are followers. Security risks in web applications are major threat today. The moving target based approach configures and shifts systems over time to increase uncertainty for the attacker. As per concept of MTD, the window of attack opportunities decreases, the cost of attack increases. There is uncertainty about the type of attacker and the author captures it as a Bayesian game where each of the agents in the game are of multiple types with respective probabilities. This is a novel approach that maps moving target defense as a Bayesian Stackelberg game.

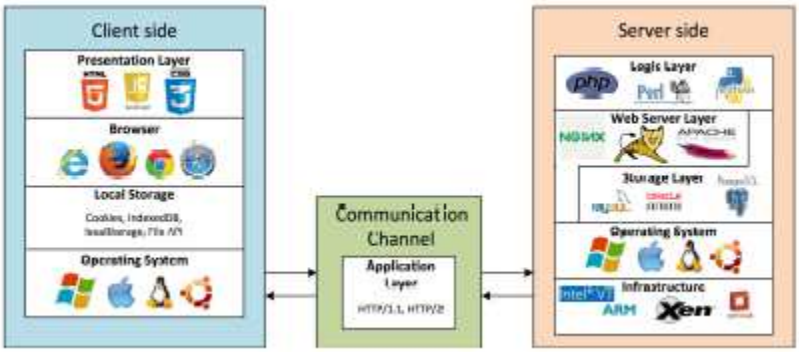


Figure 1 : A modern web application structure and its components [11]

If any layer in the architecture is compromised the system becomes unworthy. The attack can be on presentation layer on server end or malicious java script code leading to XSS vulnerability. Moving target defense will not remove the vulnerability but limit the exposure of vulnerabilities. One of the example of possible movement is that in storage

layer, running and synchronizing multiple database instances having different implementation.

“A Stackelberg game consists of a leader who commits to a strategy _rst, and then a follower adopts a strategy which maximizes its own reward based on the leader’s strategy. Here the leader has an advantage because she gets to make the first move (choose a strategy and commit)”[10]Defender in MTD is of a single type, attacker is of multiple types consistent with formulation of Bayesian Stackelberg game. Reward value for both attacker and defender are dependent on scenario.

The attacker type is dependent on following criteria:

- Difficulty of carrying out action/attack
- Popularity of target technology
- Effect of action/attack on target website
- If attacker leads to detection of attacker which depends on defender and attacker type.

To solve the game the objective is to find mixed strategy which maximizes reward for defender considering strategies of the followers. Only reward maximizing pure strategies for follower are considered for a given mixed strategy of the leader. There exists an optimal strategy for follower that is pure. The paper also shows a working example by assigning reward values to both leaders and follower. They consider two major problems, first is to find most critical vulnerability and second is to find most sensitive attacker types.

Once the defender chooses a mixed strategy, next it spends time on finding vulnerabilities and updating tables and strategies. When a vulnerability corresponding to an attack is fixed with the defender, the corresponding attacks will no longer be considered. New values are

to be calculated and compared. The removal of a vulnerability resulting in highest optimal reward for the defender can be the most critical vulnerabilities.

If the defender does not know the probability with which an attacker might attack the system. In such cases, it is important to analyze the accuracy of the table values. Overall rewards are to be recalculated using the game equation. The defender can make effort to obtain the probability values corresponding to sensitive attackers respectively.

Experimental results show that BSG strategy is optimal in nature to perform at least as good as uniform mixed strategy. The authors succeed in providing effective methodology to map moving target defense problem as a Bayesian Stackelberg game to obtain an optimal mixed strategy and hence maximizing rewards for the defender. They also provide techniques to identify most critical vulnerabilities and most sensitive attacker types to improve security that help in improving security of the web.

Most of the papers surveyed here give emphasis on research on higher levels of local autonomy and control to our defensive systems, possibly with different control and coordination interfaces. Dynamic diversity is an effective strategy for attacks involving extended sequence of requests. Adaption Engine to produce effective configurations that can be used in conjunction with random adaptions to mitigate the attacks. The major concerns common to all the papers is to combat costs for diversification, real time data collections and design of new subsystems. Few papers articulate novel algorithm to optimizatise the system and tradeoffs between cost of migration and benefits of protection. One of the interesting approach is to apply game theory approach to make more informed decision to tackle web security problem.

Live Migration

By definition, Live migration refers to the process of moving a running virtual machine or application between different physical machines without disconnecting the client or application. Memory, storage, and network connectivity of the virtual machine are transferred from the original guest machine to the destination.[12]

One of the most important applications of Live migration is proactive maintenance. Aside from this, it is used for load balancing. Live migration for moving target defense under the dynamic network category. As VMs are migration between machines, the information with attacker, about the IP and the ports are now stale.

There are two kinds of live migration with respect to virtual machines.

Pre-Copy Migration: Memory is copied before migration.

- Warm Up Phase – Hypervisor typically copies all the memory pages from source to destination while the VM is still running on the source. If some memory pages ' change (become 'dirty') during this process, they will be re-copied until the rate of re-copied pages is not less than page dirtying rate.
- Stop and Copy – VM will be stopped on the original host. The remaining dirty pages will be copied to the destination.VM will be resumed on the destination host.

If the destination fails during migration, pre-copy can recover the VM. Pre-copy retains an up-to-date state of the VM at the source during migration.

Post-Copy Migration: Source is suspended first and then the memory is copied.

- Source VM suspend – Post-copy VM migration is initiated by suspending the VM at the source. A minimal subset of the execution state of the VM (CPU state, registers and, optionally, non-page able memory) is transferred to the target.

- Pre-paging and Copy – The VM is then resumed at the target. source actively pushes the remaining memory pages of the VM to the target – an activity known as pre-paging.

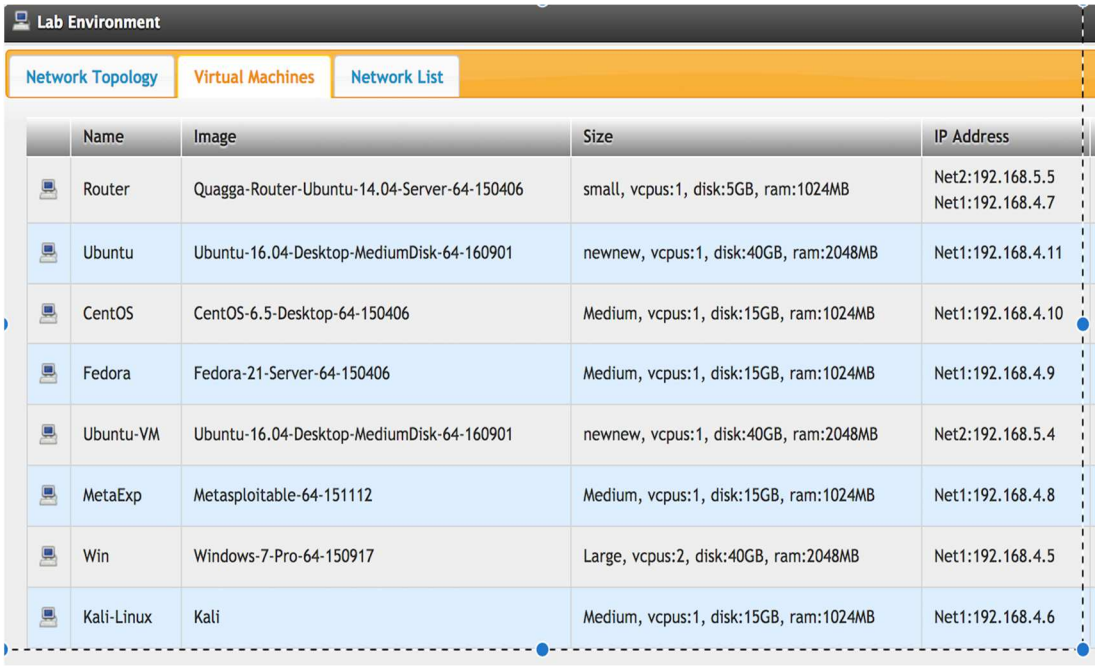
In Post-copy, the VM's state is distributed over both source and destination. At the target, if the VM tries to access a page that has not yet been transferred, it generates a page-fault. Too many network faults can degrade performance of applications running inside the VM.

CHAPTER 2

ATTACK DETECTION

Setup

I setup centralized syslog server for 6 hosts (1 Windows 7 VM, 1 Fedora 23 VM, 1 CentOS 6.5 VM, 1 Kali Linux 2.0 VM, 1 Metasploitable 2.0 VM, 1 Ubuntu 16.04 VM) in a Openstack based cloud networking environment. For this I used thoth lab, as it is easy to spin up any desired topology in it. The system details are as shown in the Figure 2.



	Name	Image	Size	IP Address
	Router	Quagga-Router-Ubuntu-14.04-Server-64-150406	small, vcpus:1, disk:5GB, ram:1024MB	Net2:192.168.5.5 Net1:192.168.4.7
	Ubuntu	Ubuntu-16.04-Desktop-MediumDisk-64-160901	newnew, vcpus:1, disk:40GB, ram:2048MB	Net1:192.168.4.11
	CentOS	CentOS-6.5-Desktop-64-150406	Medium, vcpus:1, disk:15GB, ram:1024MB	Net1:192.168.4.10
	Fedora	Fedora-21-Server-64-150406	Medium, vcpus:1, disk:15GB, ram:1024MB	Net1:192.168.4.9
	Ubuntu-VM	Ubuntu-16.04-Desktop-MediumDisk-64-160901	newnew, vcpus:1, disk:40GB, ram:2048MB	Net2:192.168.5.4
	MetaExp	Metasploitable-64-151112	Medium, vcpus:1, disk:15GB, ram:1024MB	Net1:192.168.4.8
	Win	Windows-7-Pro-64-150917	Large, vcpus:2, disk:40GB, ram:2048MB	Net1:192.168.4.5
	Kali-Linux	Kali	Medium, vcpus:1, disk:15GB, ram:1024MB	Net1:192.168.4.6

Figure 2 – System Detail

This topology is as shown in the Figure 3.

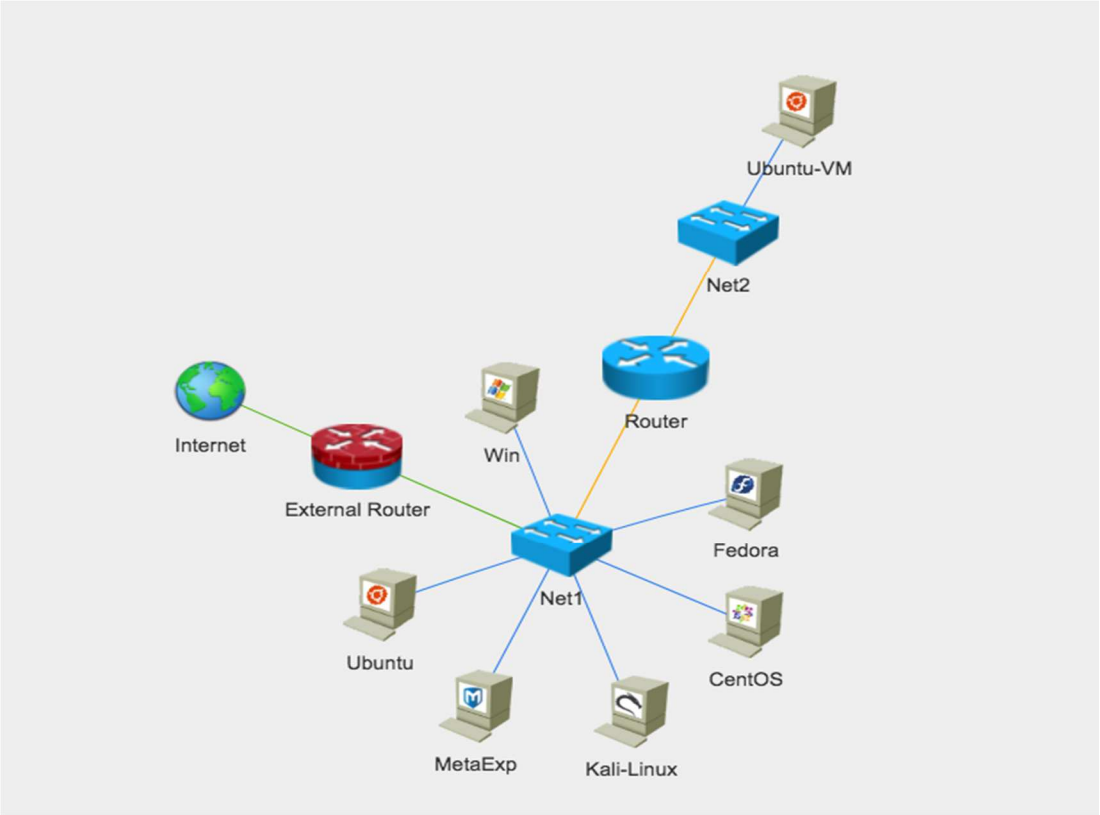


Figure 3 – Topology of the log collection system

RSYSLOG is the rocket-fast system for log processing [13]. It offers high-performance, great security features and a modular design. While it started as a regular syslogd, rsyslog has evolved into a kind of swiss army knife of logging, being able to accept inputs from a wide variety of sources, transform them, and output to the results to diverse destinations. In the topology of the Figure 1, the Ubuntu-VM is the centralized log server and rest of the machines are configured to send logs to it. This configuration is made in the rsyslog.conf file in /etc/ of the machines.

Table 2 – At Ubuntu-VM server

/etc/rsyslog.conf
provides UDP syslog reception #$ModLoad imudp #$UDPServerRun 514 # provides TCP syslog reception $ModLoad imtcp $InputTCPServerRun 514

I am using TCP for log collection as I do not want to lose out on any logs. This is shown in Table 2.On the client side , I will enable the machine to send logs to the Ubuntu-VM. Configuration to add it in mysql was also added and further connected to Log Analyzer for visualization. This has not been included in the thesis as it does not contribute to the parsing and reduction of logs.

Table 2 shows example of the Ubuntu client sending logs.

Table 3- At Ubuntu Client

/etc/rsyslog.d/50-default.conf
. @192.168.5.4:514

After setting up all the client machines as shown in Table 3, rsyslog server is restarted, so that the configuration changes are included. I could see the logs being collected immediately.

Log Collection and Analysis

The logs collected in the Ubuntu-VM are in directories with their host name. This helps in differentiating in host wise log. The format of the log is as given in the Table 4.

Table 4 - Rsyslog-LogFormat

RSYSLOG_TraditionalFileFormat
"%TIMESTAMP% %HOSTNAME% %syslogtag%%msg:::sp-if-no-1st-sp%%msg:::drop-last-lf%\n\"
May 8 18:30:11 Ubuntu-Client sudo: root : TTY=pts/11 ; PWD=/home/sammy/ftp/files ; USER=root ; COMMAND=/usr/bin/nmap -sV 192.168.4.10

Creating list of normal activates was a challenge as it required complete understanding of the system. I made a list of applications that are run on the system and categorized them as normal. This is called as audit policy.

For E.g. Every system has a defined number of CRON jobs. If the logs in the Cron.log show addition of new cron jobs or logs from a non-defined cron job, it shows abnormality. On collection of logs, the parser extracts Date, Hostname, Appname , HostIP and messages from each line of the logs. This is converted into csv as a data set.

The audit policy for the logs identifies the threat vs non threat for each log and also assign value to the feature source, to identify if the logs are generated from the system or applications. The classification of logs if threat or not ,is done by the pre defined audit

policy. This is based on the knowledge of the network hosts and its system. Few of the conditions that are considered non threatful are as follows :

- If the logs can generated from the pre defined cron jobs, they are marked safe.
- Logs showing activities from applications that are expected to be on the system
- Traffic showing pam_unix from pexec from authorized users is considered as safe.
- Changes showing useradd, groupadd , change of user id ,s sudo and root operations are marked as potential threats.
- Any unexpected process restarts, package download and user creation is marked as threat.
- Unexpected applications and their logs are considered threats, like rlogin , rsh and rexec, segfaults.
- Kernel level logs for known behavior as marked as non-threats.

The goal is to perform activates that attacker will be to perform the malicious activities in a way to evade normal signature based detection agents like Snort. Table-5 shows list of logs that were collected.

Table 5 - Types of Logs Collected

S.No.	Log File Name	Description
1	CRON.log	Cron job logs which are scheduled to automate repetitive tasks
2	NetworkManager.log	Logs activities of network manager daemon which provides detection and configuration for systems to connect automatically to network
3	Root.log / su.log	Activities performed by root user
4	sudo.log	Activities performed by users that are also sudoers
5	Rsyslog	Standard event logging from the host activities configured by rsyslog utility
6	Groupadd.log	Logs activities when group of user changes
7	rlogin.log	Captures rlogin service activities
8	Rsh.log and rexec.log	Captures non interactive programs run on remote server using rsh
9	gnome-session.log	Captures activities from gnome -desktop environment
10	kernel.log	Has up to date errors and status updates while booted in operating system.
11	dhclient.log	Logs messages from dhclient
12	systemd.log	Logs all systemd actions

The procedure followed to create the dataset is shown in the Table 6

Table 6 - Procedure to create dataset

Creating Dataset

Input : Rsyslog Directories from Central Log server
 	
Output : CSV file with features
 	
noiseList : List of log attributes that can be eliminated from the dataset
 	
SourceMap : Maps appname to System or Application
 	
while !EndofLogfiles do
 	
Spilt log lines to extract attributes from Log line
 	
if(attributes) do not match noiseList
 	
CSV.add(attributes)
 	
end if
 	
end while
 	
for all attribute in CSV file
 	
if attribute{appname} in SourceMap
 	
CSV.append{attribute}(Source=SourceMap(attribute.appname))
 	
end if
 	
end for
 	
for all attribute in CSV file
 	
Input the value for threat value in Boolean T/F
 	
end for

Results

Once the logs are filtered as threatful and non threatful, all the threatful logs are combined together. The time stamp field is converted into EPOCH time and all logs are sorted according to the time. This gives us the complete picture of the connections made by the systems to one another in a time line. This can be a ground for detection of attack. This approach can be used as a starting step to detect attacks like Advanced Persistent Attacks that are designed to pass the detection of Network Intrusion System like Snort.

The Table 7 shows subset of events that form time line of abnormal activities.

Table 7 - Sample filtered log lines of threatful events

Epoch Time	Source IP	Service	Destination IP
1494289681	192.168.4.11	sudo	192.168.4.8
1494293780	192.168.4.11	vsftpd	192.168.4.11
1494294548	192.168.4.11	sudo	192.168.4.8
1494304744	192.168.4.8	rlogind	192.168.4.11
1494304793	192.168.4.8	rshd	192.168.4.11
1494304812	192.168.4.8	rlogind	192.168.4.11
1494304901	192.168.4.8	rshd	192.168.4.11
1494305111	192.168.4.8	rlogind	192.168.4.11
1494305196	192.168.4.8	rlogind	192.168.4.11
1494305756	192.168.4.8	rshd	192.168.4.11
1494305787	192.168.4.8	rshd	192.168.4.11

In the above table only source IP and destination IP, service and epoch time are included. The message field is truncated in this example.

CHAPTER 4

DOCKER MIGRATION

Setup

I set up a topology of 4 system in Thoth lab [15]. All four system has same configuration. The detailed configuration of the machines is show in the Figure 4

Name	Image	Size	IP Address
VM2-Big	Ubuntu-16.04-Desktop-BigDisk-64-160901	m1.medium, vcpus:2, disk:40GB, ram:4096MB	Sw1:192.168.1.2 Sw2:192.168.2.6
Vm4-Big	Ubuntu-16.04-Desktop-BigDisk-64-160901	m1.medium, vcpus:2, disk:40GB, ram:4096MB	Sw2:192.168.2.5
VM3	Ubuntu-16.04-Desktop-MediumDisk-64-160901	newnew, vcpus:1, disk:40GB, ram:2048MB	Sw1:192.168.1.6
VM1	Ubuntu-16.04-Desktop-MediumDisk-64-160901	newnew, vcpus:1, disk:40GB, ram:2048MB	Sw1:192.168.1.5

Figure 4 - Configuration of Machines for Live Migration

The VM 1 is source machine, where the containers are running. VM 3 is machine that I consider as destination. This machine is where I will transfer my container state. It has same configuration as VM 1. VM 4 is machine which acts as external client to test connections to internal VM1 and VM3. The VM1 and VM3 form a DMZ and can be accessed on through VM2- Big. This machine has Apache2 installed that allows it to become proxy for internal machines. Figure 5 shows the topology.

On VM1 and VM3, I have installed Docker 1.13 CE edition along with experimental flag turned on. This is a mandatory step as it allows live migration [13]. It can be enabled by editing the docker.conf file like below:

```
# cat /etc/systemd/system/docker.service.d/docker.conf
[Service]
ExecStart=
```

ExecStart=/usr/bin/dockerd -H fd:// --experimental=true

After that the docker details will look like below

#docker version

Server:

Version: 1.13.0

API version: 1.25 (minimum version 1.12)

Go version: go1.7.3

Git commit: 49bf474

Built: Tue Jan 17 09:50:17 2017

OS/Arch: linux/amd64

Experimental: **true**

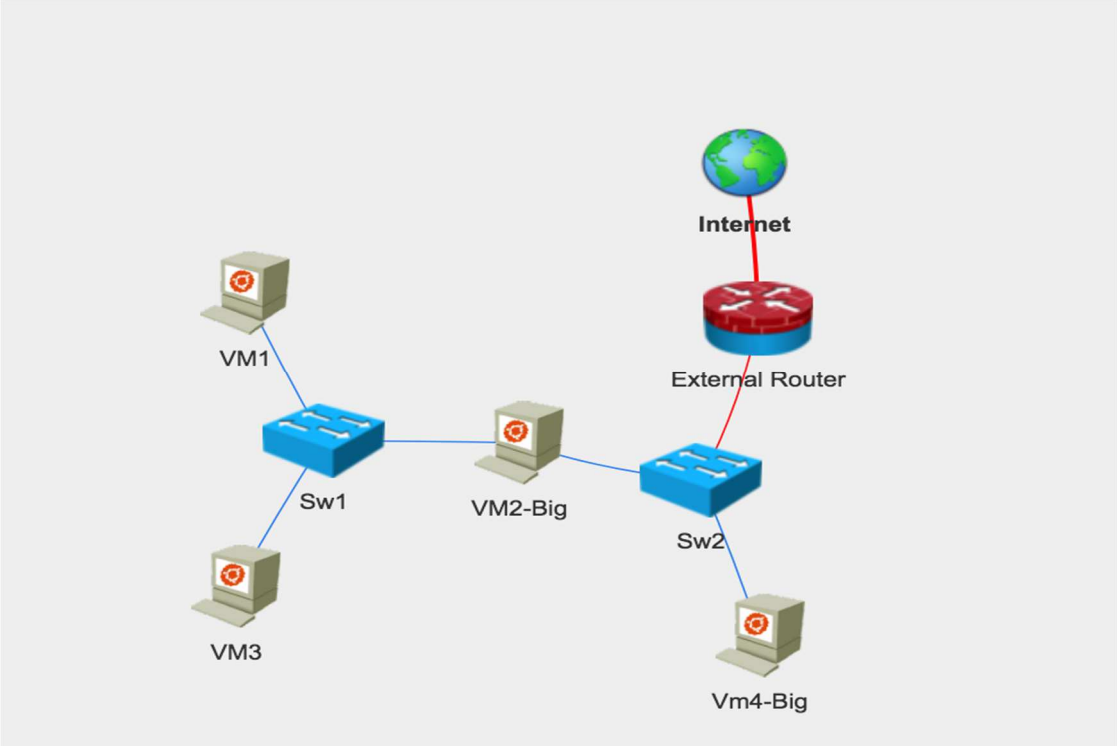


Figure 5 - Toplogy for live migration

I restart the docker service to make sure all changes are reflected.

Another important tool is CRIU (Check point restore). I built it using the source from github (https://github.com/xemul/criu). It has several dependencies and there are to be present before the source is installed (make, make install). They are: libnet1-dev git build-essential libprotobuf-dev libprotobuf-c0-dev protobuf-c-compiler protobuf-compiler python-protobuf libnl-3-dev libpth-dev pkg-config libcap-dev asciidoc.

To check if CRIU has been installed properly, the CRIU check command should return the following output.

criu check

Warn (criu/autofs.c:79): Failed to find pipe_ino option (old kernel?)

Looks good.

On VM 2 – Big, Apache2 is used as a proxy. The initial rewrite rule in 000-default.conf is that all traffic is transferred to VM1. On migration, from VM1 itself , automatically the rewrite rule will be change to point all traffic to VM3.

I established a passwordless SSH between VM1 – VM3 and VM1 – VM2 for live migration and IP redirection.

In order to establish that the Hardware and Operating system of the system is not affected by the attacker, a Hardware/Operating system trust system can be used to ensure authenticity. This will ensure the kernel level security and the solution proposed in the thesis is used for application level security.

Methodology

I take a sample container from docker hub called tutum/apache-php [16]. It is a Base docker image to run PHP applications on Apache. To run this docker container I use the following command on VM1

#docker run –name = test -d -p 80:80 tutum/apache-php

Now a http apache server has started running on port 80. This can be confirmed by accessing http://localhost from the browser on VM1.

Before adding the checkpoint, the commit of docker container is required.

#docker commit test

The next step is to add a check point. The checkpoint directory I use is /tmp/ for simplicity. Any custom path can be given. Basically, it will store all the images of the state in this folder. This is done via running command like below

#docker checkpoint create test cr1 --checkpoint-dir=/tmp/

Now that a check point is created, the next step is to transfer the cr1 state information to VM3. I am using secure copy and the secure copy is automated using passwordless ssh.

#scp –r /tmp/cr1 192.168.1.6:/tmp/

Now we have the state information on the VM3. Before we resume the container state on VM3, there is a additional step that has to be performed. I do this in the setup. I create the same image of apache-php on VM3 and keep it in checkpoint ready state. The reason to this is to save time in creating new image on the go and also to have a same root file system for the container.

Once the VM3, is in checkpoint restore ready state , I use the below command to restore the container. This command is run from VM1 to VM3

#ssh 192.168.1.6 ‘docker start –checkpoint cr1 test –checkpoint-dir=/tmp/’

The next step is to change the ipaddress redirection to move the traffic from VM1 to VM3. This change is done from VM1 itself. I use the following command to change the ip-redirection

ssh 192.168.1.2 ‘| sudo -S sed -i "s/RewriteRule \/\.* http\:\/\/192\.168\.1\.5\/ \[R\]/RewriteRule \/\.* http:\/\/192\.168\.1\.6\/ \[R\]/g"' /etc/apache2/sites-enabled/000-default.conf

The time between docker commit to IP redirection together is the downtime of the application in the docker. I will evaluate the down time in the next section.

Performance Evaluation

I evaluated the performance of the container migration by using the previously mentioned apache-php image. The experiment was performed 10 times for every 1000 request and trend was plotted on the average of these experiment. The graphs in this section show timing for each of the steps in migration and the traffic.

Number of requests vs Time to commit (seconds):

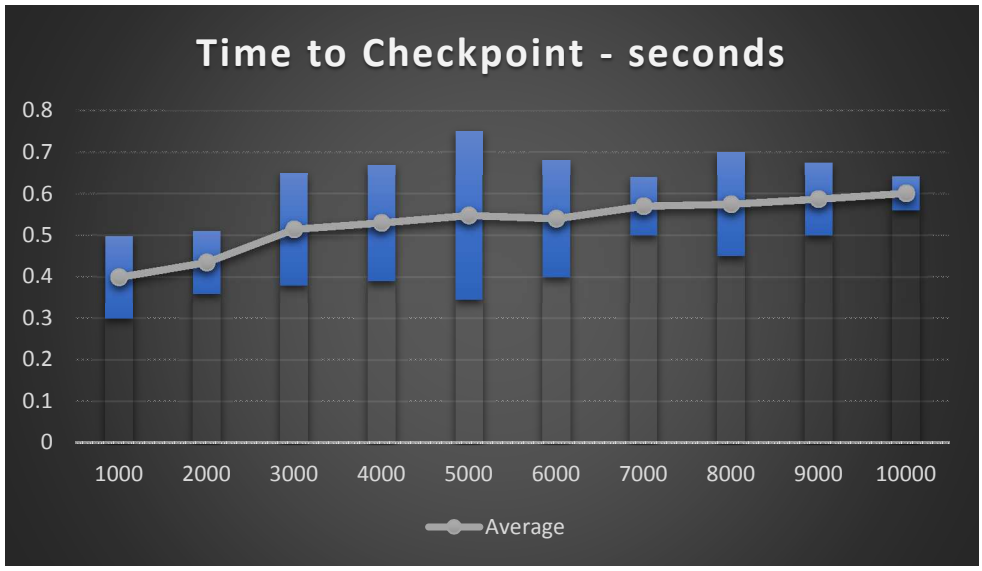


Figure 6 - Number of Requests Vs Time to Commit

Average time to commit is 0.53 seconds

Number of requests vs Time to checkpoint (seconds):

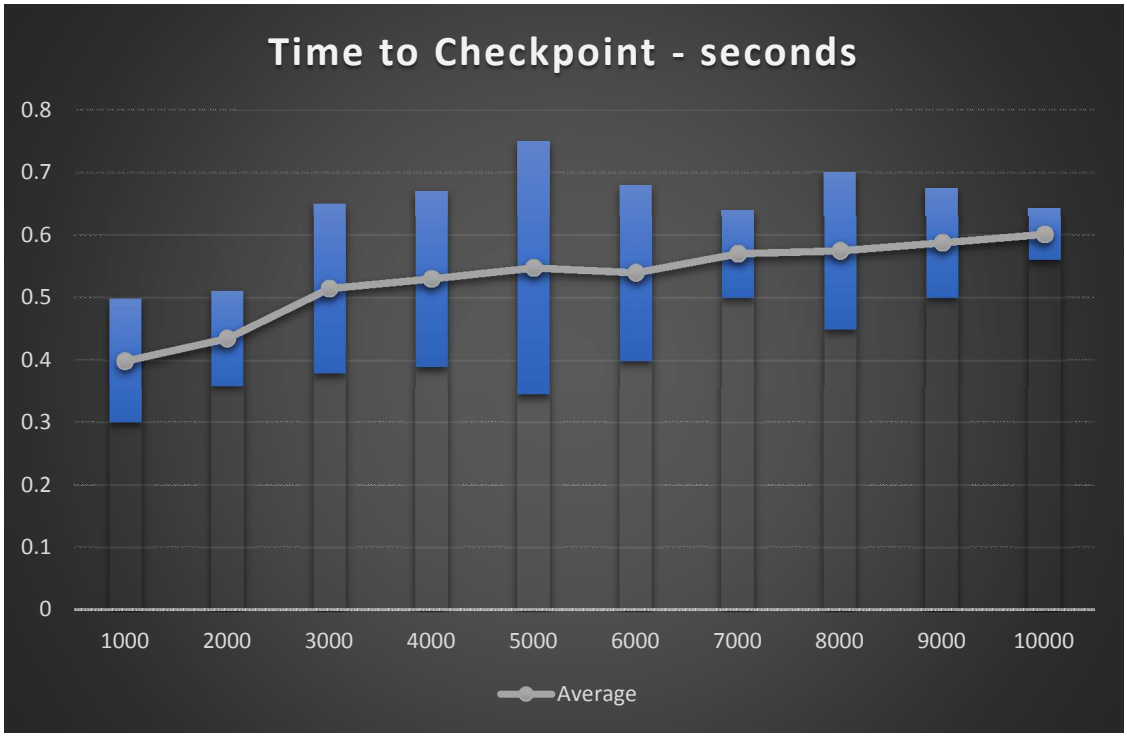


Figure 7 - Number of Requests Vs Time to Checkpoint

Average time to check point is 0.59 seconds

Number of requests vs Time to transfer (seconds) :

Figure 8 - Number of Requests Vs Time to Transfer

Average time to transfer the state is 1.69seconds

Number of requests vs Time to restore(seconds)

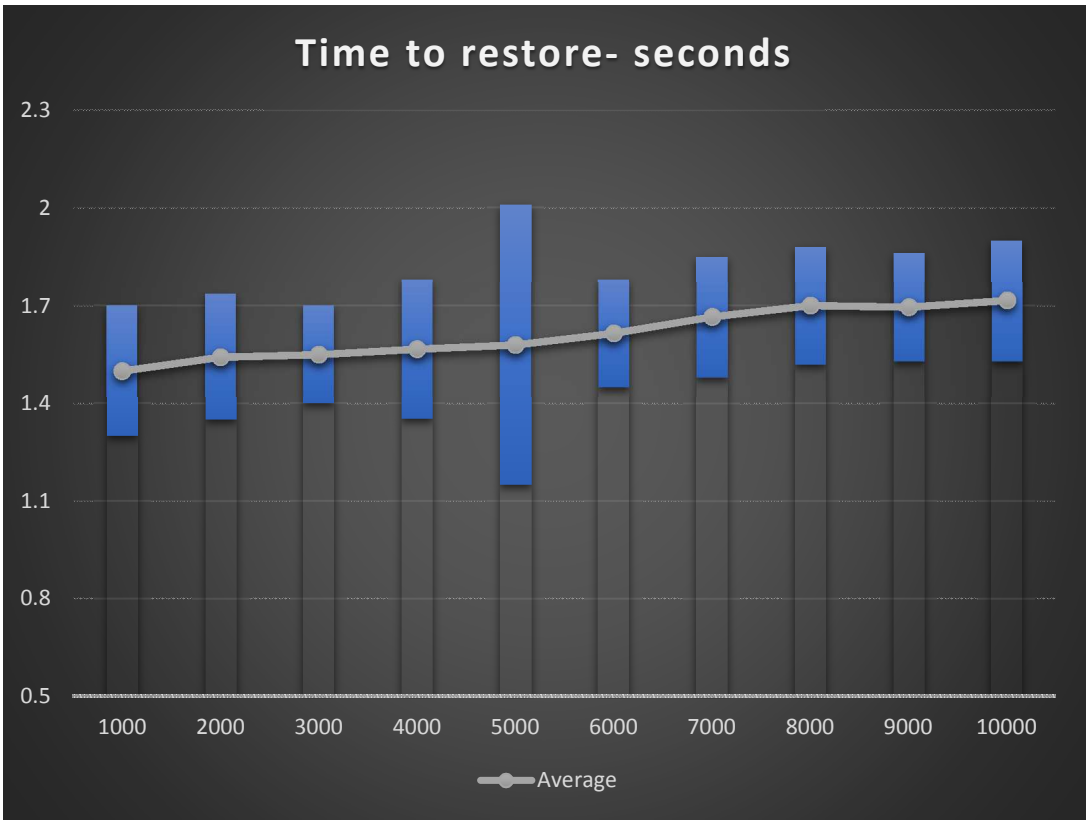


Figure 9 - Number of Requests Vs Time to Restart

Average time to restore the container is 1.61 seconds.

The total average time to migrate a container is **4.30** seconds.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, I was able to trace events for attack detection and also show a proof of concept for live migration of docker containers. The average time to migrate the container is 4.30 seconds for 80mb size of state.

This can be improved by adding pre-copy type of element like virtual machine migration. I am closely following the docker project and I wish to improve the setup as and when docker new features are introduced.

I also want to suggest an extension of event tracing by temporal logic so that it gives a differentiation between connection and control. In that way, it can be used to detect Advanced Persistent Threats.

The bigger picture here is that as we moving towards light weight containers, migration can definitely an important feature to use.

REFERENCES

[1] Arijs, P. (n.d.). The Container Monitoring Blog. Retrieved May 30, 2017, from https://www.coscale.com/blog/docker-usage-statistics-increased-adoption-by-enterprises-and-for-production-use

[2] "CSD-MTD | Homeland Security". Dhs.gov. N.p., 2017. Web. 29 May 2017.

[3] Okhravi, H., Rabe, M., Mayberry, T., Leonard, W., Hobson, T., Bigelow, D., & Streilein, W. (2013). Survey of cyber moving targets. Report Massachusetts Inst of Technology Lexington Lincoln Lab MIT. LL-TR-1166.

[4] Carvalho, M., & Ford, R. (2014). Moving-target defenses for computer networks. IEEE Security & Privacy, 12(2), 73-76.

[5] Evans, D., Nguyen-Tuong, A., & Knight, J. (2011). Effectiveness of moving target defenses. In Moving Target Defense (pp. 29-48). Springer New York.Chicago

[6] Zhuang, R., Zhang, S., Bardas, A., DeLoach, S. A., Ou, X., & Singhal, A. (2013, August). Investigating the application of moving target defenses to network security. In Resilient Control Systems (ISRCS), 2013 6th International Symposium on (pp. 162-169). IEEE.

[7] Jia, Q., Sun, K., & Stavrou, A. (2013, July). Motag: Moving target defense against internet denial of service attacks. In Computer Communications and Networks (ICCCN), 2013 22nd International Conference on (pp. 1-9). IEEE.Chicago

[8] Han, Y., Lu, W., & Xu, S. (2014, April). Characterizing the power of moving target defense via cyber epidemic dynamics. In Proceedings of the 2014 Symposium and Bootcamp on the Science of Security (p. 10). ACM.

[9] Hong, J. B., & Kim, D. S. (2014, June). Scalable security models for assessing effectiveness of moving target defenses. In Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on (pp. 515-526). IEEE.Chicago

[10] Debroy, S., Calyam, P., Nguyen, M., Stage, A., & Georgiev, V. (2016, February). Frequency-minimal moving target defense using software-defined networking. In Computing, Networking and Communications (ICNC), 2016 International Conference on (pp. 1-6). IEEE.

[11] Vadlamudi, S. G., Sengupta, S., Taguinod, M., Zhao, Z., Doupé, A., Ahn, G. J., & Kambhampati, S. (2016, May). Moving target defense for web applications using bayesian stackelberg games. In Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (pp. 1377-1378). International Foundation for Autonomous Agents and Multiagent Systems.

[12] "Live Migration". En.wikipedia.org. N.p., 2017. Web. 29 May 2017.

[13] "Rsyslog". Rsyslog.com. N.p., 2017. Web. 29 May 2017.

[14] Makam, Sreenivas. "Docker 1.13 Experimental Features". Sreenivas Makam's Blog. N.p., 2017. Web. 30 May 2017.

[15] "Thoth Lab". Thothlab.org. N.p., 2017. Web. 30 May 2017.

[16] Hub.docker.com. N.p., 2017. Web. 30 May 2017.

[17] "CRIU". criu.org. N.p., 2017. Web. 29 May 2017.